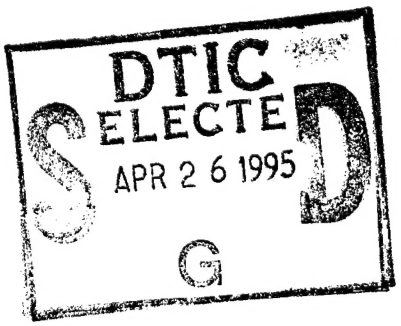
Adaptive Differential Visual Feedback for Uncalibrated Hand-Eye Coordination and Motor Control

Martin Jägersand and Randal C. Nelson

Technical Report 579
December 1994



UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

Adaptive Differential Visual Feedback for Uncalibrated Hand-Eye Coordination and Motor Control *

Martin Jägersand, Randal Nelson
Department of Computer Science, University of Rochester, Rochester, NY 14627
{jag,nelson}@cs.rochester.edu
http://www.cs.rochester.edu/u/{jag,nelson}/

December 14, 1994

Abstract

We propose and implement a novel method for *visual space* trajectory planning, and adaptive high Degree Of Freedom (DOF) visual feedback control. The method requires no prior information either about the kinematics of the manipulator, or the placement or calibration of the cameras, and imposes no limitations on the number of degrees of freedom controlled or the number of kind of visual features utilized. The approach provides not only a means of low-level servoing but a means to integrate it with higher level visual space trajectory and task planning. We are thus able to specify and perform complex tasks composed of several primitive behaviors, using both visual servoing and open loop control, where the number of sensed and controlled signals varies during the task. We report experimental results demonstrating a factor of 5 improvement in the repeatability of manipulations using a PUMA arm when comparing visual closed-loop to traditional joint level servoing. We also present experiment statistics showing the advantages of adaptive over non-adaptive control systems, and of using redundant visual information when performing manipulation tasks. Finally, we demonstrate usefulness of the approach by using it to specify and execute complex tasks involving real-world robot manipulation of rigid and non-rigid objects in up to 12 degrees of freedom. The manipulation is performed in the context of a semi-autonomous robot manipulation system.

*Support for this work was provided by ONR grant N00014-93-I-0221, Sverige-Amerika Stiftelsen and the Fulbright Commission

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	technical report

4. TITLE AND SUBTITLE
Adaptive Differential Visual Feedback for Uncalibrated Hand-Eye Coordination ...

5. FUNDING NUMBERS
N00014-93-I-0221

6. AUTHOR(S)
Martin Jägersand and Randal C. Nelson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES
Computer Science Dept.
734 Computer Studies Bldg.
University of Rochester
Rochester NY 14627-0226

8. PERFORMING ORGANIZATION

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESSES(ES)
Office of Naval Research
Information Systems
Arlington VA 22217

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
TR 579

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution of this document is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
(see title page)

14. SUBJECT TERMS
robot control; adaptive control; visual feedback control; hand-eye coordination; real-time active vision; robot user interfaces

15. NUMBER OF PAGES
28 pages

16. PRICE CODE
free to sponsors; else $2.00

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	UL

1 Introduction

Robot manipulators with sensing capabilities restricted to sensing joint angles are limited in the kinds of behavior they can exhibit. Essentially, they can execute only a preprogrammed sequence of movements. Yet, with little variation, this is how robots currently used in industrial applications perform their tasks. One reason for this simple approach is the difficulty encountered when trying to handle complex sensing systems within the analytic modeling frameworks traditionally used in robotics. In typical industrial settings, complex sensing systems such as vision, if employed at all, are used only off-line when the system is in a well specified state. For example, a camera mounted on a manipulator (eye-in-hand) has been used to locate parts on a table, but the process is performed only when the camera is in a specific, pre-specified position with respect to the work surface. During movement to pick up a part located by the vision system, sensory input from the (now moving) camera is not used. This approach simplifies modeling of the mapping between image and global world coordinate systems, but represents a rather limited use of visual sensing, and is somewhat lacking in robustness. Continuous feedback provides a more robust approach to sensory control, and recent work has begun to develop approaches for utilizing both sophisticated visual and somatic (e.g. force and torque) sensors in this way.

In this paper we propose and implement a novel method for *visual space* [1] trajectory planning, and adaptive high Degree Of Freedom (DOF) visual feedback control. Our approach is inspired by previous work in differential feedback, but differs in that, while previous work has concentrated only on the visual servoing algorithm, we integrate visual servoing with higher level visual space trajectory and task planning. We want to perform complex tasks composed of several primitive behaviors, using both visual servoing and open loop control, where the number of sensed and controlled signals varies during the task. To do this, we need a controller capable of handling a wider variety of situations than is possible using previous approaches. We present an adaptive controller based on, and integrated with, a visual space planning scheme, that enables this higher-level, general task specification. We experimentally evaluate its performance, and then illustrate its usefulness in several complex real-world manipulation tasks.

Differential visual feedback control in robotics was pioneered in the eighties, primarily by Lee Weiss and Arthur Sanderson [1, 2, 3]. They proposed a classification of different types of visual control, and performed experiments in simulation with differential visual feedback controllers. Since then, real world implementations have been made by a number of researchers, e.g., [9, 10, 4, 5, 6, 7, 13, 19, 15, 16, 20, 22]. For a review of this work we direct the reader to [17] or [18].

The adaptiveness of the controller is a key to the success of our approach. Previous approaches have either assumed that the system need only be calibrated once (e.g. [9]) using a set of specific "test movements", that it can be decoupled into a set of single variable adaptive controllers (e.g. [3]), or that the system can be modeled using an ARMAX model (e.g. [5]). These systems can only model a small class of visual-motor transfer functions. Limitations include requiring the transfer function to be near linear over the desired operating space, restricting the number and type of visual features that can be used simultaneously, and limiting the number of cameras and/or camera placement.

In order to avoid some of the above problems, we found it helpful to look outside the mainstream of traditional control theory. In particular we drew inspiration from numerical analysis, specifically, the Broyden class of optimization methods for nonlinear problems (a survey can be found in [33]). Basically, we implement a Jacobian estimator of the full local visual-motor transfer function that updates the local model [2]

[1] Visual space is most easily thought of as (but not limited to) a space of image (pixel) coordinates defining location and orientation of objects we manipulate, taken from one or more cameras watching the scene

[2] Systems based on differential visual feedback are often advertised as "uncalibrated" or "model free". Of course they have some internal model while running. What is meant is that they somehow learn (or identifies) these models rather than requiring prior (analytic) models and exact calibration.

using only information from the naturally occuring process noise. This suffices for accurate feedback control over highly non-linear global spaces. We also use this local model, learned while moving under feedback control, to execute open loop movements while, for instance, visual features are occluded during an insertion movement.

At the higher level end of task specification and trajectory planning, we have been inspired by the user friendly man-machine interfaces found in [28, 29, 30]. Our ultimate aim is to provide natural, low bandwidth human interaction with our system, combining the advantages of autonomous operation and telemanipulation. In contrast to Ikeuchi and Kuniyoshi, we neither center our planning around a prior world model nor do we make use of a global world coordinate frame. Instead, all specification takes place in image space, either directly, via operator pointing in the image, or with the aid of more sophisticated image processing techniques such as automatic feature location or object recognition.

The paper is organized as follows: The next two sections focus on the differential visual feedback controller. Section 2 describes the controller we have implemented. In section 3 we present extensive experimental tests of the controller performance, some of which are applicable to a wide variety of visual servoing algorithms, showing and that visual servoing improves repeatability by a factor of 5 on a PUMA robot arm, that the adaptive controller outperforms a fixed gain controller. We also demonstrate good convergence properties for a 6 DOF controller, and show that using redundant visual information (i.e. tracking many more visual features than the number of DOF's we control) improves positioning, yielding subpixel accuracy in visual space.

The rest of the paper focuses on issues of visual space task specification and planning at a level above the differential visual feedback controller. In section 4 we describe how we represent tasks, and present an extended controller architecture capable of handling a wide variety of robot manipulation problems via visual space task specification and planning. Section 5 shows how visual space task planning is used to solve real world manipulation problems. First we show how to compose different modes of control to produce an insertion primitive. Then we use the system to solve a toddler-level puzzle, putting different shaped pieces into corresponding slots. Finally we have two PUMA robot arms learn to manipulate a flexible piece of foam held between their end effectors. Using only images of how the foam is to be positioned and deformed, the controller has to learn to perform the manipulations that achieve the goal while simultaneously controlling all 12 joints in the two arms. Section 6 describes how our system can be applied in a more general setting, using visual as well as other sensors, through what we call a *Perceptual Action* task level control scheme.

2 Differential visual feedback

This section describes the differential visual feedback controller we use, and issues relevant to its design.

The situation depicted in fig. 1 shows a typical setup for visual servoing. For three and higher DOF control we typically want to use at least two cameras, spaced widely enough apart to give well conditioned depth cues. Otherwise, camera placement is fairly arbitrary. In the images taken by the cameras we extract, and continuously track, the positions of interesting features, for instance, the position of the robot end effector. We denote the vector of visual features by $\mathbf{y}$. [3] Features suitable for visual control can be drawn from a much wider class of measurements in the image, and we will discuss this later. However, for the purposes of the derivations in this section, thinking of the visual features as image the location of tracked features will suffice.

The vector $\mathbf{x}$, represents the output signals from the controller. In this example, for conceptual simplicity, we let $\mathbf{x}$ be robot joint positions. In our actual implementation we will use a combination of position and

[3] We use capital letters to indicate matrices, bold face for vectors, and plain for scalars and functions.

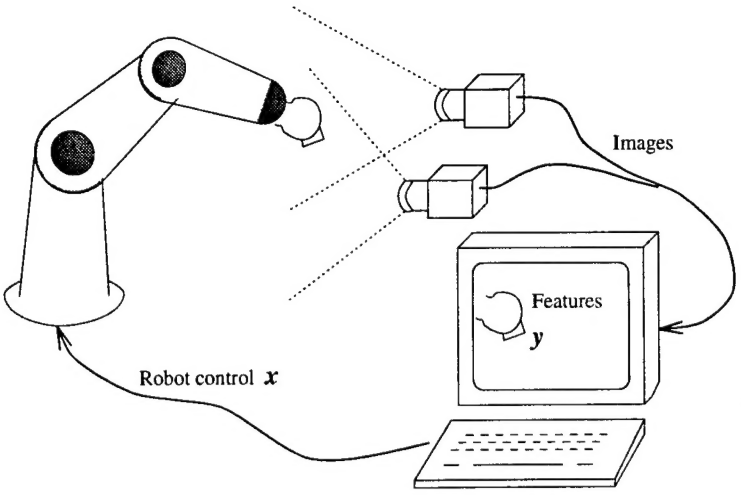


Figure 1: Typical visual control setup uses two cameras placed so that they can observe the workspace from two different viewpoints. Otherwise, placement is arbitrary, and the controller has no prior knowledge of the camera locations, their relation to the robot, or the robot kinematics. The setpoints for the robot joint angles are specified by the vector $\mathbf{x}$, and visual perceptions, or features are represented by the vector $\mathbf{y}$, These are related by an initially unknown transfer function $\mathbf{y} = f(\mathbf{x})$.

velocity control.

In a traditional visual control setup we have to know two functions, either a-priori, or through some calibration process. The first is a camera, or vision calibration function h that transforms image space feature locations to a Cartesian world coordinate system (typically having its origin at the base of the robot). The second is the robot kinematic calibration function g that transforms desired robot end effector positions, given in world coordinates, into joint angles. The final robot positioning accuracy depends on the accuracy of both these functions ($f^{-1} = g(h(\cdot))$), since feedback is only performed over the joint values.

It was realized early on that accuracy could be improved by using residual visual error, measured after a movement was executed, to perform a correction move. [2] classified such schemes as *Position based (iterated) look-and-move* approaches. The next step was to notice that if goal points were provided in image coordinates and realized through joint angles, then the intermediate world coordinate system could be completely eliminated. Weiss calls this *Image based look-and-move.* Finally, if instead of supplying visual error only occasionally, our system servos joints using a continuous visual signal, we have what Weiss calls *Image based visual servoing.*

We present a system which can learn a sufficing mapping from image based (vision space) specification, to robot (joint space) execution primitives. The system requires no prior models of the transfer function; it learns the model while performing the task without explicitly introducing any extra learning steps or movements. It is capable of learning multivariate (coupled, vector valued) transfer functions with an arbitrary number $|\mathbf{y}| = m$ of visual inputs and an arbitrary number $|\mathbf{x}| = n$ of output control signals. The method places only very loose constraints on which transfer functions can be successfully learned.

2.1 Derivation of a control method

In general, an unknown visual-motor transfer function $f : \Re^n \rightarrow \Re^m$ can be written as a multidimensional Taylor expansion. In particular we are interested in the linear approximation:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + J(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + R_2(\mathbf{x}_0, \mathbf{x})$$

where for a sufficiently "smooth" f the residual term R_2 is small for $\mathbf{x} \approx \mathbf{x}_0$. J is called the visual motor Jacobian and is defined as

$$(J_{j,i}) = \frac{\partial f_j}{\partial x_i}$$

The visual motor Jacobian relates small changes in vision space to small changes in motor (control) space, e.g., for $\mathbf{y} = f(\mathbf{x})$, $\mathbf{y}_0 = f(\mathbf{x}_0)$, $\Delta\mathbf{y} = \mathbf{y} - \mathbf{y}_0$ and $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$ we have:

$$\Delta\mathbf{y} \approx J\Delta\mathbf{x}$$

Thus the visual motor Jacobian acts as a local model of the transfer function behavior around $\mathbf{x}_0$ (and $\mathbf{y}_0$).

Given that we observe an initial perceptual state represented in the feature vector $\mathbf{y}_0$ and that we wish to bring about the (goal) state $\mathbf{y}^*$, the best prediction using our linear model J, of the required change $\Delta\mathbf{x}$ in robot control signals, is given by solving the system of equations:

$$\mathbf{y}^* - \mathbf{y}_0 = J\Delta\mathbf{x}$$

Executing this move, i.e. setting $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}$ changes the perceived state to $\mathbf{y}_1$. Unless f is a linear transfer function, $\mathbf{y}_1$ is typically not equal to $\mathbf{y}^*$. However, if $\|\mathbf{y}^* - \mathbf{y}_0\|$ is small, and f smooth, $\|\mathbf{y}^* - \mathbf{y}_1\|$ is likely to be smaller than $\|\mathbf{y}^* - \mathbf{y}_0\|$, and the process can be repeated yielding a sequence $\{\mathbf{y}_2, \mathbf{y}_3 \ldots \mathbf{y}_k\}$ successively closer to $\mathbf{y}^*$. What we have is a quasi Newton method for solving a set of nonlinear equations in $\mathbf{x}$ (find $\mathbf{x}^*$ such that $0 = \mathbf{y}^* - f(\mathbf{x}^*)$). In a continuous setting we have a similar control law:

$$\mathbf{y}^* - \mathbf{y} = KJ\frac{d}{dt}\mathbf{x}$$

where K is a gain matrix.

These two control laws implement the *image based look-and-move* and the *image based visual servoing.* These or similar control laws have been used in previously published work e.g. [9, 15, 20]. However, there are two compelling reasons why the above design is not sufficient. First, it is well known that Newton methods are not globally convergent [34]. Even in cases where the continuous system should be stable, the low sampling frequency of a TV camera based vision sensory signals makes it difficult to obtain both stability and reasonable performance in the discrete time version ([7]). At the expense of generality in the visual processing, some researchers (e.g. [13]) have circumvented this problem by using special purpose high speed "vision" systems. Drawing on results in optimization we instead use a globally convergent step restricted method, adapting the step length to keep it within a range where our local model is sufficiently accurate. In spirit, this is similar to the control theory approach of changing controller gains and limits dynamically to account for local conditions in different parts in the work space.

The second reason is that, in a high DOF system, when the simple control scheme servos on a single, distant goal, it may drift substantially due to differences in model accuracy along different directions in the high DOF space. Relying only on a distant goal may also result in unacceptable deviation from the desired path should some unexpected disturbance occur. The solution to this problem involves the generation of *way points*, which are essentially additional visual goals describing the visual situation at intermediate points. By requiring a system to pass near or through such way points, drift from the desired trajectory can be

eliminated. The current *low level visual space trajectory generator* lays out way points along straight lines in $\Re^m$, where m is the dimension of of the visual perception space. The distance between the way points is adapted to fulfill a Marquart condition of model accuracy. The Marquart algorithm is simple [34]: If the model error at time k is $d_k = \mathbf{y}_k^{measured} - \mathbf{y}_k^{predicted} > d_u$ adjust the step size downward, for instance halve it. If $d_k < d_l$, then increase the step size. The tuning parameters d_u and d_l are set differently for different types of movements (e.g. long range transportation v.s. fine manipulation), trading accuracy for time.

The implicit assumption in planning trajectories only along straight lines is that, if more complex trajectories are required, they will be provided by the higher level modules. Exact straight line trajectories are typically not achievable, but for reasonably short distances in visual space a straight line plans have proven to be a good enough approximation to achieve convergence and stability.

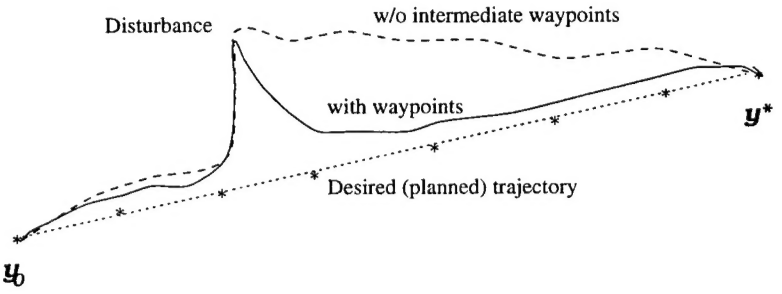


Figure 2: Planning intermediate way points in visual space reduces trajectory error compared to pure servoing on the goal.

Dividing the transition between $\mathbf{y}_0$ and $\mathbf{y}^*$ into several subsegments, with intermediate visual goals along the straight line between them also allows us to bring the manipulator back onto the desired trajectory as soon as possible in the event of a deflection, by servoing on the nearest subgoal between the current pose and the goal. This is illustrated in fig. 2. This behavior also reflects how the human arm reacts to a deflection.

2.2 Model estimation

We have not yet addressed the issue of where the local model Jacobian $J = J(\mathbf{x})$ comes from. One approach is to use geometric models, derive an analytical expression for the Jacobian, and calculate a numerical Jacobian at each sampled point $\mathbf{x}$. This approach has been used for traditional world-joint space manipulator kinematics in cases where analytic inversion is hard. An advantage is that a rough model can be used, depending on the feedback, to yield high precision positioning. Other approaches can be described as partially adaptive, being applicable only to a restricted set of visual motor transfer functions, requiring partial (analytic) modeling of the system (e.g. restricted to systems suitable for ARMAX modeling [5, 8], or using only image location visual measurements under a weak perspective assumption [16]).

We want to solve manipulation problems that are unsuitable for analytic modeling at all, such as the manipulation of flexible foam in 12 DOF shown in section 5.3. We also want to place the fewest possible restrictions on the class of transfer functions we can model. This rules out systems depending on near linear transfer functions in the desired operating space (e.g., [9, 15]), and methods that would be inefficient for high DOF systems (e.g. [22] where calculating a Jacobian update requires nm measurements, with nm being the product of the number of sensory and controlled signals)

We use a *Jacobian model estimation and updating* scheme based on the information obtained more or less for free while performing the task. After making the movement $\Delta\mathbf{x}$ and observing the change $\Delta\mathbf{y}_{measured}$ in

the feature values, we wish to modify the Jacobian to bring our model into agreement with the measurement, i.e. to make $\Delta \mathbf{y}_{measured} = J_{k+1}\Delta \mathbf{x}$. In general (for $size(J) > [1,1]$), this problem does not have a unique solution. One choice is to make the minimal change necessary. In this case

$$\hat{J}_{k+1} = J_k + \frac{(\Delta \mathbf{y}_{measured} - J_k \mathbf{x})\mathbf{x}^T}{\mathbf{x}^T\mathbf{x}}$$

which can be easily seen to satisfy our model agreement condition.

An appealing and intuitive way of visualizing how the updating works is to consider it in a coordinate frame aligned with the last movement. Let P be a coordinate change matrix from the original basis O for $\mathbf{x}$ to an ON basis O' in which has the first basis vector aligned with the movement $\Delta \mathbf{x}$ we last made. Using P and $P^T = P^{-1}$ we can freely transform the measured change in visual appearance $\Delta \mathbf{y}$ and the motor-visual Jacobian J back and forth between the two coordinate bases. The formulas $\Delta \mathbf{y}' = P^T \Delta \mathbf{y}$ and $J' = P^T J P$, transform the objects into O' and the inverse formulas bring them back.

In the course of a particular move, we obtain a difference approximation to the directional derivative of the transfer function along the direction of the movement. In the basis O', this difference approximation, $\frac{\Delta \mathbf{y}'}{\Delta x_1}$, is aligned with the first basis vector in O' since $\Delta \mathbf{x}' = (\Delta x_1, 0, \ldots, 0)$. In the Jacobian this derivative is represented in the first column. Thus to incorporate the new derivative information $\frac{\Delta \mathbf{y}'}{\Delta x_1}$ into the Jacobian, we simply insert it into the first column, leaving the information along the other directions (in the other columns) unchanged. Looking at the general updating schema above in the O' frame, it is clear that this is exactly what happens there also (just consider that the fact that $\Delta \mathbf{x}' = (\Delta x_1, 0, \ldots, 0)$). Thus our updating schema fulfills the minimum change criterion.

In the practical applications we also perform adaptive weight filtering in the updating to suppress noise in the estimate. The new model J_k is obtained from the formula $J_k = a\hat{J}_k + (1-a)J_{k-1}$, where $a = min(1, \frac{\|\Delta \mathbf{y}\|}{Ndi})$ and Ndi represents our current model confidence given as the maximum allowable step length. The estimation technique we have described falls into a class called Broyden methods [33] used in nonlinear optimization.

2.3 Implementation issues

If J were a square, full rank matrix, a unique $\Delta \mathbf{x}$ could be found for each $\Delta \mathbf{y}$ by Gaussian elimination. However, J is generally neither square, nor full rank. In fact, as we will see later, it is an advantage in a real system to have many more visual feature values than controlled DOF's giving an over-determined system. Several methods exist for finding approximate solutions $\Delta \mathbf{x}$, to such a system under various optimality criteria (e.g., normal equations or SVD based methods). We use the numerically stable QR factorization method, projecting a least-squares optimal solution onto the control space.

Mapping this Jacobian based control scheme onto a real robot involves a number of other issues, related to the low level control of the robot arm. The most direct approach is to use a discrete look, then move strategy, where we wait for each incremental move to be completed before initiating the next cycle. This method is satisfactory for short, high-precision moves, where the precision obtainable by extracting visual features in a stationary scene is needed. For long moves, however, this process very slow. It also results in uneven movement which uses significant excess energy, produces extra wear and tear on the robot, makes prediction difficult in interacting systems, and is aesthetically unpleasing.

What we do in this case, is to utilize a more sophisticated control structure for the robot, that instead of providing only discrete moves to a specified position, allows asynchronous updating of a joint position "set point" to which the robot servos under some simple control law (e.g. damped proportional control). On the Utah/MIT hand, an appropriate set point controller comes as part of the hardware. On the Puma, the set

point controller was implemented through primitives provided by RCCL [31]. By running the visual control loop fast enough to keep the set point just ahead of the robot, we are able to achieve smooth trajectories by essentially "leading" the robot around. By varying the lead distance of the setpoint, and velocity and acceleration parameters we get a mixed mode controller having some of the advantages of both discrete and continuous control modes.

We run the vision feature tracking and visual feedback algorithm on a distributed network of Unix workstations. The worst case delays due to Unix process scheduling and Ethernet communication contention are, from a control systems viewpoint, extremely long. The setpoint method has the advantage, in this case, over a simple velocity servoing schema. Even if sensory input is completely shut out, all that happens is that the robot goes to the next setpoint, and comes to a rest, as in the look and move approach. In more usual situations we switch from a smooth servoing mode during transportation phases, through an intermediate phase, which slows down the robot somewhat during visual measurements to increase their accuracy, to a pure iterated look-and-move behavior for the highest precision moves.

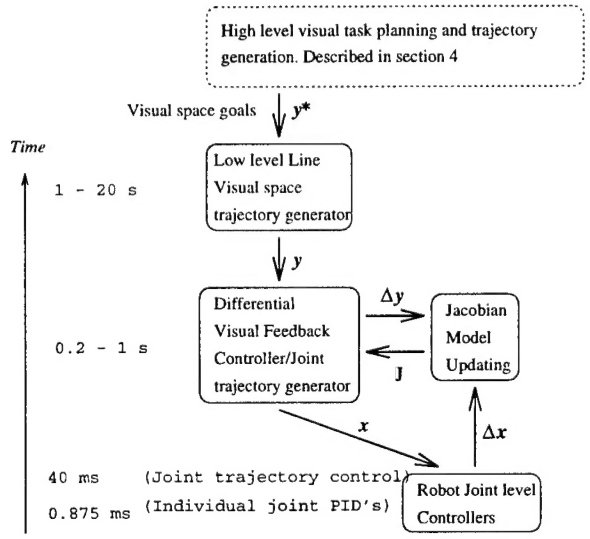


Figure 3: Structure of the adaptive differential visual feedback control algorithm, and the time frames each part runs in. Arrows between modules indicate direction of information transfer.

The structure of the control algorithm we have implemented is shown in Fig. 3. In addition to the modules shown, there are real-time visual feature trackers that provide information about the current visual state. The system is implemented as a pvm [32] distributed program running vision processing, and visual space control routines on a 8 processor SPARCserver 2000 multiprocessor, and robot control on a SPARCserver 330.

3 Experiments with the Adaptive Control Method

Surprisingly, a thorough experimental evaluation of the accuracy related properties of differential visual feedback control has never, to our knowledge, been performed. The closest approach is [13], who showed for a 2 DOF implementation, the advantage of visual feedback over open loop control when model errors are large. Chen et al in [21] show that two heavy industrial robot arms can perform a peg-in-hole parts mating task with a clearance of 0.7mm between the peg and the hole.

In this section we describe some of the characteristics of our adaptive control method that makes it particularly suited to the kind of visual space specification and control we do. First, we evaluate repeatability under visual feedback control, and compare the results to standard joint level control. Then we demonstrate how a 3 DOF adaptive controller is significantly better at dealing with the typically large model errors of a uncalibrated vision system than a fixed gain controller. A 6 DOF controller is generally even more severely affected by model errors. In a second experiment, with a 6 DOF system, we show that as long we stay within reasonable error bounds in our linear model we typically achieve convergence to 0 pixel error in image space. Finally we motivate the importance of using redundant perceptual information by tracking many more features than the degrees of freedom under control, showing that absolute (world) positioning accuracy improves for a visual processing front end with fixed accuracy on tracking individual features.

3.1 Experimental Procedure

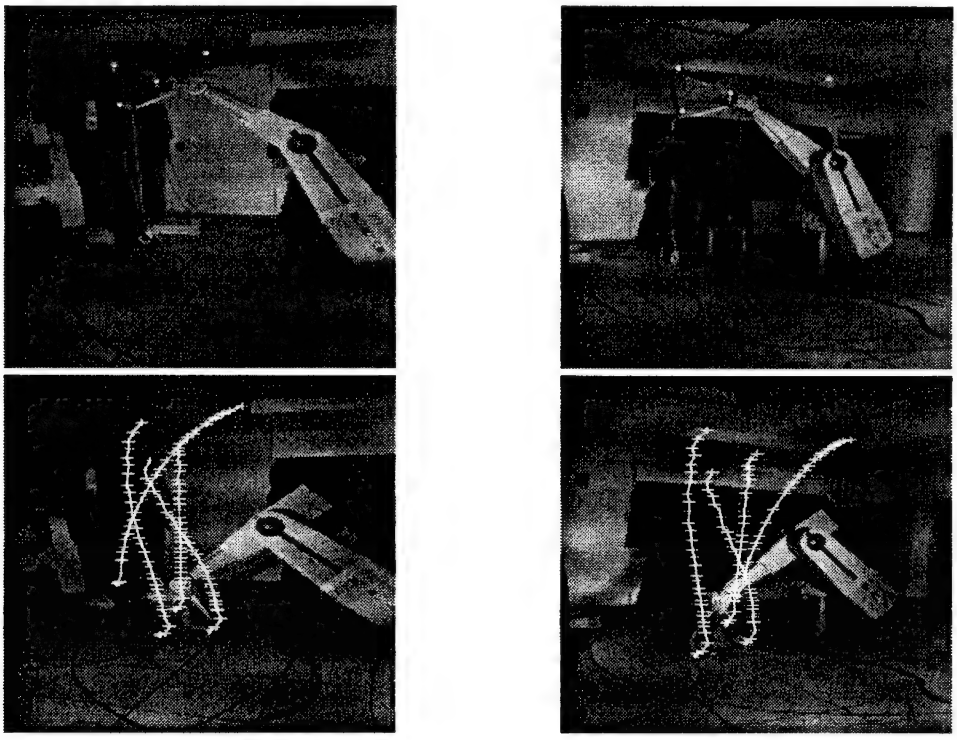


Figure 4: Experimental setup for controller experiments. Above: Initial configuration of robot as seen in the left and right cameras. A visual goal near the floor is given to the algorithm. Below: Goal configuration achieved by the controller. The image space trajectory of the tracked features is overlaid.

The following experiments use a common setup. Two cameras are positioned to view the desired portion of the robot workspace. We have found that the exact placement is not particularly important, and that as long as the angular separation of the cameras is greater than about 30 degrees we have a sufficiently well conditioned problem. On the robot end effector we have mounted a 30 by 40 cm T-shaped foam piece with three small light bulbs, one at each end. A fourth light bulb is connected to a rod extending 40 cm above, and 40 cm out of the plane of the T. We use the same general feature trackers that we use later in the visual specification experiments. The light bulbs simply make the tracking more reliable and accurate.

The main purpose of these experiments is to study kinematic accuracy. To make sure the trackers keep up with the features, and that the test rig does not vibrate very much, we run the robot at low speed. At the start of an experiment series a Jacobian approximation is obtained by executing a sequence of test moves near the middle of the (visually defined) workspace.

A typical experiment is performed as follows: The robot is moved to a pre-defined pose specified by joint angles. Readings are taken from the trackers to define a visual goal. This ensures that the visual goal is actually attainable, something which is typically not true of a set of randomly selected feature values. After registering the goal, the robot is deflected to a random position in the selected workspace. During the deflection all control algorithm input is shut out, disabling model updating. At the end of the deflection additional noise is injected into the Jacobian model, and other adjustments are made to the control algorithm depending on the purpose of the experiment.

The controller is then turned back on, and relevant measuring processes are started, acquiring, for example, timing and visual trajectory information of the sort plotted in fig. 4. The controller is turned off when the visual goal is achieved within a preset accuracy, or when it is no longer moving any closer to the goal. Endpoint accuracy measurements are taken at this point. Then the whole procedure is repeated a number of times (between 50 and 200 times) until results are statistically significant.

3.2 Repeatability

We tested repeatability under closed loop visual control and compared the results to traditional joint control, both with our own experiments and with published figures. Visual feedback control overcomes those problems in traditional inverse kinematics approaches where positioning accuracy is limited by the accuracy of the kinematic model. We can also improve positioning performance by reducing the effect of some non-kinematic disturbances, such as gear backlash and manipulator flexibility. We did not explicitly measure accuracy. However, since no additional calibration errors are introduced for visual feedback control, when specifying the goals in the same visual space, it can be argued that visual control accuracy should be similar to repeatability.

To make the measurements, we mounted a 0.001" accuracy dial meter on the robot end effector, allowing us to make very precise position measurements. Watching a semiconductor junction from the side on standard LED's mounted on the robot end effector provided us with very accurate visual positions. As a reference surface, we use a piece of glass mounted on a very heavy tripod. To sample different parts of the robot workspace, we positioned the tripod in different locations and orientations. Repeatability was measured by moving the robot on random trajectory towards the glass measuring surface until the dial meter was in contact with the glass, reading the dial value and visual or joint values, and then retreating on a random trajectory, before trying to reachieve the visual or joint goal. The algorithm is specially designed to execute the final approach from a different, random direction each time.

The cameras are placed close to the end effector for high visual accuracy. One pixel image offset represents about .25mm in the real world. The measurements are made in 1 DOF, and roughly along the direction of joints 1 and 4 (neither of which are gravity loaded). Joints 1-4 are driven, but they are coupled so only 3 DOF's are controlled in the robot. The experiments were performed with the arm near full extension.

Table 1 shows our results for visual feedback repeatability. The results indicates convergence to subpixel accuracy. We think "vernier accuracy" effects from the high dimensional ($m = 12$) input space can account for this. Compared to joint control repeatability measured under the same conditions, visual space repeatability is about 5 times better. The figure published by Unimation for the PUMA 762 joint control repeatability is three times better than ours. We can only hypothesize why, since Unimation does not give any details of their experiment. One hypothesis is that our robot is just old and wear on the mechanics account for our

Measured (mm)			Specified (mm)
Visual feedback	With perfect visual alignment	Joint feedback	PUMA manual
0.13	0.079	0.64	0.2

Table 1: Measured repeatability of our Unimation PUMA 762 robot under visual and joint feedback control, compared to figures given in the PUMA manual

higher error. Another is that Unimation measures same-path repeatability, i.e. the robot reaches the goal point the same way both times in a measurement, while we measure different path repeatability. Same path measurements tend to cancel out many sources of error, and can explain the better figure. We measured same-path repeatability on our arm to be about .12 mm. The precision limit imposed by the joint angle encoders is also .12mm.

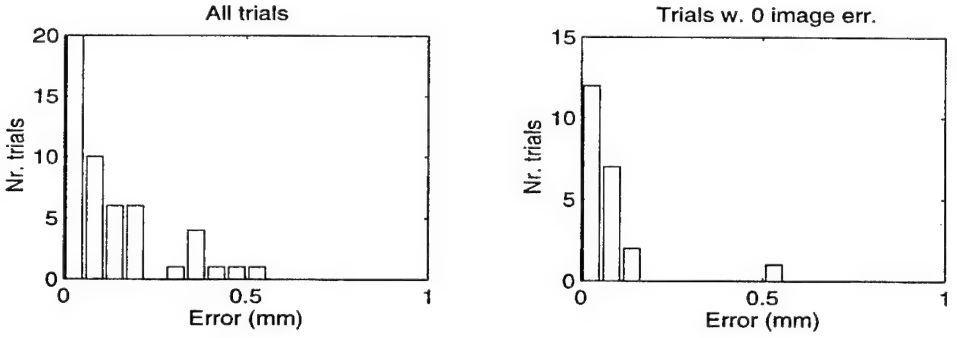


Figure 5: Left: Distribution of errors in closed loop visual feedback experiment. Right: Showing only the trials which converged to zero visual (image) error

22 of 50 trials converged to 0 visual error. The average positioning error on these is much better than on the whole data set. We believe the main reason why more trials did not converge is due to the difficulty in positioning the manipulator, near or below its resolution limit. Very close to the goal, noise in the controller causes it to drive the motors randomly back and forth. This samples a lot of possible positions below the resolution limit, and by stopping if/when we hit 0 visual error, we can achieve positioning below the resolution limit. The error distribution of 50 repeatability trials is shown in fig. 5.

The distribution of positioning errors for the open loop experiments can be seen in fig. 6. The two big modes in the distribution are most likely due to the different backlash errors introduced by driving joint j1 in one of two directions during a measurement. The visual feedback and open loop repeatability experiments were both performed using only the small workspace visible in the cameras. To investigate whether joint control repeatability changed when doing bigger movements, we varied the trajectory length. We did not find any dependence on trajectory length. We don't see any reason why this should not hold for visual feedback control as well.

3.3 Adaptive v.s. Non-adaptive control

In this experiment we compared performance of adaptive and non-adaptive controllers in 3 DOF. The basic experimental plan was to add varying amounts of noise to the Jacobian model, and then compare trajectory and endpoint errors for the adaptive and non-adaptive cases. The disturbed Jacobian is defined by the

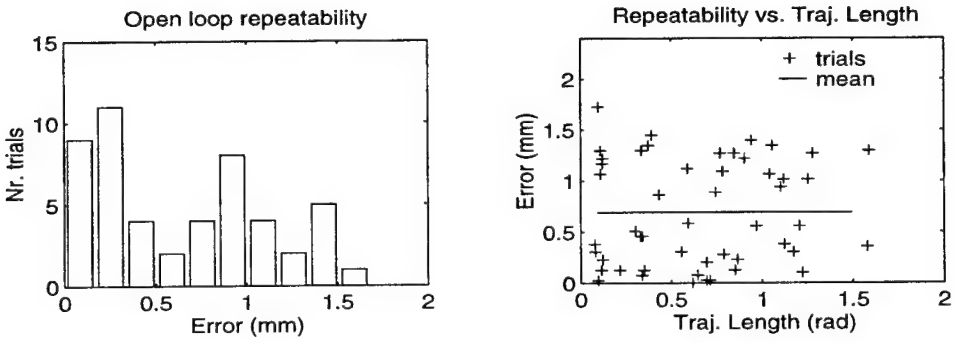


Figure 6: Left: Distribution of errors under standard joint control only. Experiment otherwise identical to the visual feedback one. Right: Varying the length of the (random) trajectory does not affect joint control repeatability

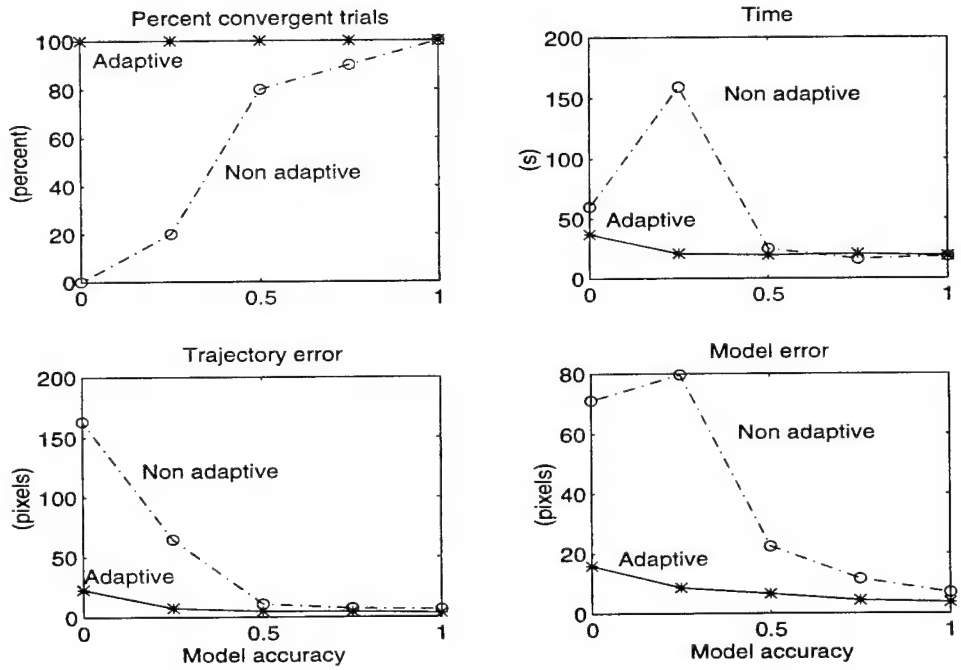


Figure 7: Results from 50 runs of an experiment injecting varying amount of random noise into the Jacobian, thus disturbing the internal model in the controller

equation

$$J_{dist} = a * \hat{J} + (1 - a) * J_{random}$$

where $a \in [0, 1]$ indicates the model accuracy and J_{random} is a random $m \times n$ matrix with elements drawn from a uniform distribution on the interval $[-|2\hat{J}_{i,j}|, |2\hat{J}_{i,j}|]$. This restricts the gain, but not the direction (angular transformation) of the random Jacobian. 50 trials were run with varying values of the model accuracy parameter.

In fig. 7, we see that for the non-adaptive algorithm, the percentage of the trials which converged to the correct endpoint (within $\|\mathbf{y}_{final} - \mathbf{y}^*\| \leq 2pixel$ in this case) quickly falls to zero for model accuracies a below 0.5. The adaptive algorithm has no problem, even with completely random initial Jacobians.

We define trajectory error as the average deviation in visual space from the planned straight line between the initial feature values $\mathbf{y}_0$ and the goal $\mathbf{y}_*$. Model error is the average difference between the predicted move and the actual move for each sampling interval $|\Delta\mathbf{y}_{pred} - \Delta\mathbf{y}_{measured}|$. The difference between model and trajectory error is that trajectory error does not penalize errors in gain along the desired trajectory, whereas model error does. On the other hand if the robot starts with a large initial error in position, the trajectory error will accumulate until the robot is back on the desired trajectory whereas model error will only penalize for incorrect predictions.

For random models (a small) both model and trajectory error increases somewhat for the adaptive algorithm. This increase can be accounted for by a few early erroneous movements. These noisy movements quickly update the Jacobian along the corresponding directions, and the error is reduced to near zero. In theory, a number of independent moves equal to number of controlled DOF should suffice to determine the Jacobian exactly, but because the measurements are noisy we use variable gain filtering to reduce its effect. This increases the robustness of the updating procedure, but slows down the convergence somewhat. For the non-adaptive algorithm, we find a huge increase in both trajectory and model error. This is caused by divergent trials, where the robot goes off in a completely incorrect direction. The peak in both time and model error for a around 0.25 is caused by long period oscillatory behavior at the transition between convergence and divergence. An illustration of a typical oscillatory move is shown in fig. 8.

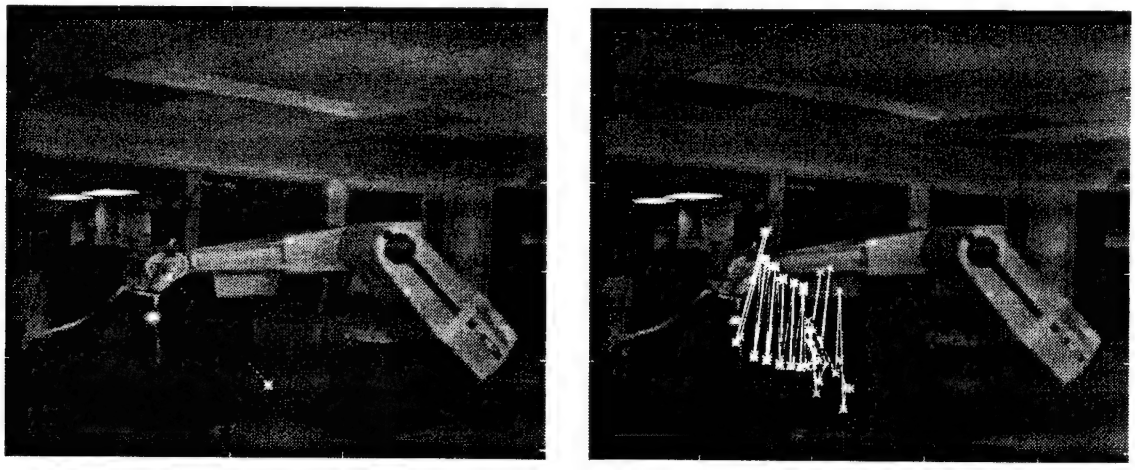


Figure 8: Left: Visually planned path. Right: Typical oscillatory behavior of non-adaptive controller when model error is near the convergence limit.

3.4 6 DOF adaptive control

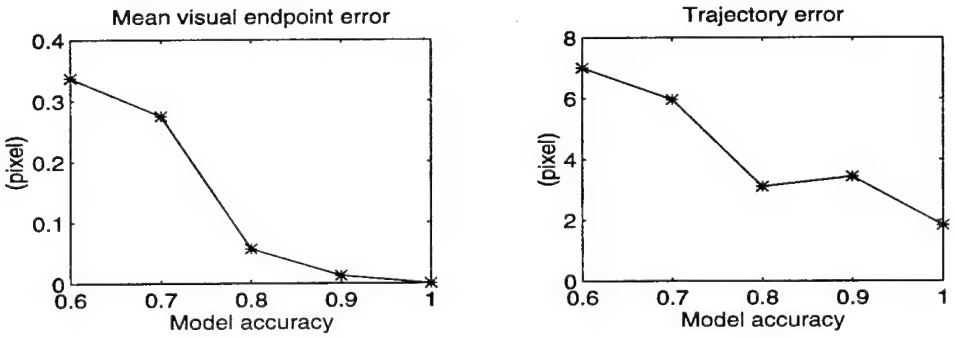


Figure 9: Average final endpoint error/feature (left), and deviation from straight line trajectory in $\Re^{16}$ feature space (right) for a 16 tracked features and 6 controlled DOF problem

When controlling the full 6 DOF of the manipulator we do not get quite the same robustness against model errors as in the 3 DOF case. Model errors worse than .6 yield a significant and increasing proportion of divergent trials. To get near 100% convergence we needed to restrict a to be .6 or above. In a test of 50 trials in this range, two did not converge. One showed oscillatory behavior, and the other diverged immediately. In this experiment we ran the controller to its convergence limit rather than halting at a fixed accuracy. A remarkable 100% of the trials with the best model achieve zero final visual error. This is down to 20% when the model error is 0.6. In general, average visual endpoint errors as well as trajectory errors remain low even at low model accuracies, as can be seen in fig. 9.

3.5 Effects of Over-determined Input Spaces

In this experiment we studied how accuracy is affected by adding redundant visual information, making the system more or less over determined. We found two main advantages. The first is that errors due to inaccuracies in either visual feature tracking or goal generation can be lessened. In the best case, if we add m signals with zero mean σ^2 variance noise, (admittedly not typically a valid assumption in a vision system), the variance, and thus the noise of the final signal decreases as $\frac{\sigma^2}{m}$). However, even if this model of the signal does not hold, as long as at least some of the visual errors are uncorrelated we will get an improvement from perceiving redundant visual features. In general, constraining the robot movement to the smaller n-dimensional action space effectively implements averaging along the directions in which we have redundant measurements. The second main advantage is that in the event of one or more of the features becoming occluded, or giving a badly conditioned measure, we can rely on the others to solve the task. We can even add more features dynamically, first letting the system identify their motor visual transfer function, without using them for control, then putting them into the control loop.

Figure 10 shows the effects of increasing the number of tracked points. In the fist experiment we placed the cameras 0.5m from the robot, allowing accurate feature tracking. We introduced varying amounts of error in the visual goal specification by adding a vector in a random direction, with a magnitude uniformly distributed in the range 0.0 to 1.5, and 0.0 to 3.0 pixels (expected value 0.75 and 1.5 pixels) to each of the visual goals. Final positioning accuracy was measured with the dial meter as in the repeatability experiment. The results indicate that for inaccurate visual goals, positioning improves significantly when using redundant visual information.

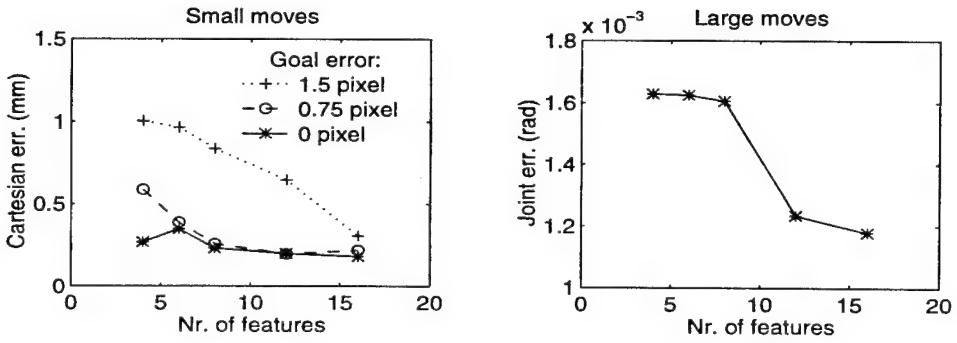


Figure 10: Left: 250 runs varying error in the visual goal specification, and the number of visual features used shows that particularly for imprecise visual specifications having redundant features help. Right: A similar experiment but having imprecise tracking by placing the cameras very far away shows a similar result. Differences in joint errors are small, but statistically significant

In the second experiment we placed the cameras 6m from the robot, thus decreasing the accuracy in tracking. Here it is more practical to measure errors as differences in joint encoder values. We observe a statistically significant improvement in joint space end point accuracy when increasing the number of features. The trajectory error, however, increases from about 1.5 pixels when using 4 visual features to 3 pixels when using 16. This is because the straight line trajectories requested by the trajectory generator are generally not achievable with a large number of points. In practice this is not a problem, even though our low level visual planner operates only with straight lines, because we seldom need to make long movements under high DOF control. The high DOF control is typically used only for fine manipulation in a small space close to a goal, where a straight line trajectory is a good approximation of actually achievable configurations.

4 Visual space task representation, planning and control

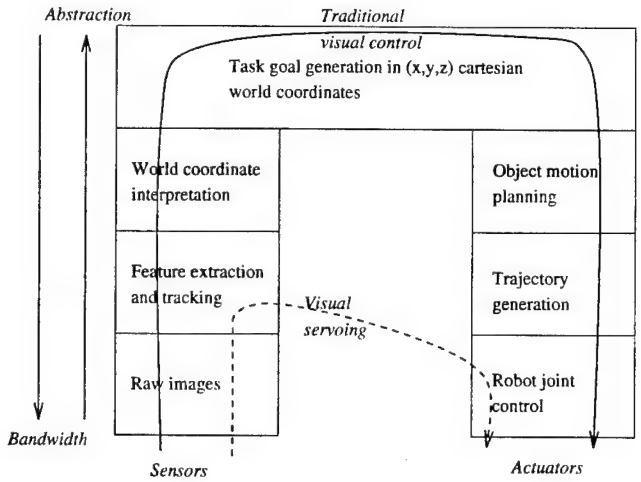


Figure 11: The general structure of a traditional visual control system, and the "short cut" of the visual servoing approach. Adapted from Corke 94

In the past visual feedback control has been promoted primarily as a method for bypassing hard calibration problems related to robotics in general and visually guided robotics in particular. The situation is graphically represented in 11. The present work retains some of the above flavor, and we present results involving the visual control of real devices with more degrees of freedom, and more complex kinematics (e.g. highly elastic beams) than has previously been reported. However, we want to shift the main focus to emphasize instead the suitability of the visual (and other sensory) reference frames for task specification, particularly when compared to Cartesian world coordinate frames.

A traditional robot has its program stored in terms of a sequence of points expressed in world (or sometimes joint) coordinates. Using this sequence as a program yields behavior that is completely inflexible to changes in the environment. In order to be adaptive to different environments a system needs perceptions of the environment. The traditional bare robot perceives only its joint configuration, which is an exceedingly sparse representation of most aspects of the environment, and one that has little relevance to many important tasks.

To be relevant, a representation should efficiently encode the goals of a task in terms that can be realistically derived from available sensors. Those goals can be visual in nature, for instance, the appearance of the final product is an overall goal in the case of an assembly task. They can also be formed from other sensory signals, for example, the torque with which a bolt is tightened (and where it is inserted) may be a goal for a subtask of the assembly task. The crucial observation is that goals are more naturally coded as patterns in sensory signals, (perceptions) than as motor positional configurations.

In order to to achieve such goals, we need either explicit transformations between the sensor spaces and motor control space, or control strategies which can adapt to provide direct control on very general classes of sensory inputs. The former approach often requires excessive amounts of calibration. We adopt the latter, and extend the differential visual feedback controller developed in the previous sections to carry out general visually specified tasks. We call the resulting system a *perceptual action* strategy, emphasizing that we control motor actions directly from high level perceptions.

4.1 Representation levels in visual space specification

The representations used in a *perceptual action* visual specification and control system using visual goals are very different from those in a traditional control system centered around the world coordinate reference frame. The central representations in a perceptual action, system as shown in fig. 12 are perception or feature vectors $\mathbf{y}$. From one perspective, that of the feedback controller, these are just vectors of real numbers. From a task perspective, however, the vectors need to capture the parts of the system state we are interested in manipulating, and be well conditioned with respect to the task we are performing. In other words the perceptual goals have to be good attractors.

One way of looking at the situation is to view robot control as an optimization problem of transforming the initial perceptual state, denoted $\mathbf{y}_0$ to the goal state $\mathbf{y}^*$. In the perceptual action paradigm, the perception vector space is both the space of specification, and the space in which the controller operates. Visual servoing, is an attempt to solve the optimization problem by gradient descent in this space. Closing of the feedback loop allows servoing to zero perceptual visual error without accurate calibration needed in a traditional system servoing only in robot joint space. The convergence of the algorithm clearly depends on the qualities of the measurements.

The perception vectors $(\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_k)$ contains all the necessary information to describe the task. The adaptive visual feedback controller we presented in the previous sections, without initial knowledge about f, provides an effective method of finding a control signal sequence $(\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_k)$, that efficiently takes the system through the states $(\mathbf{y}_i)$, while at the same time learning about f.

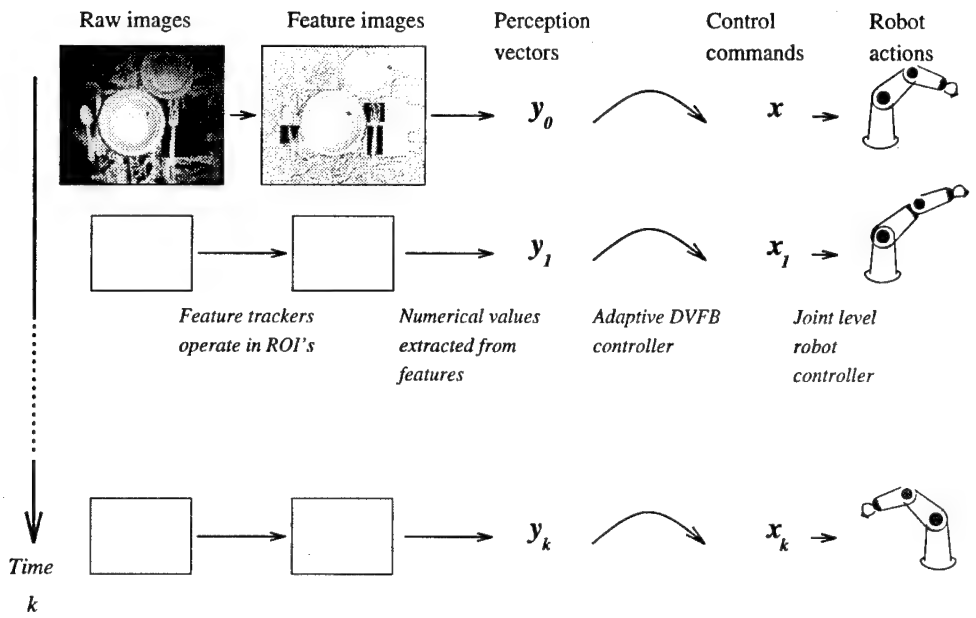


Figure 12: The representation levels in a Perceptual Action system. The main level is the Perception vectors. Goals are specified as perception vectors, and when moving they represent all state information relevant for attaining the goal

In the important case of a perception vector based only on vision, each perception vector ($\mathbf{y}_i$) corresponds to a measure in some image $\mathbf{y}_i = g(Image_i)$. The idea is that $\mathbf{y}_i$ is chosen to be a fingerprint of $Image_i$. The fingerprint should be unique, but only in a very weak sense. It should assign the value $\mathbf{y}^*$ only to images of world configurations which fulfill our goal, and values close to $\mathbf{y}^*$ only to images which are close to the goal. Thus if we have a sequence of goals we can think of the sequence $(\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_k)$ as a sequence of goal images, although in reality they are measurements of the image. This is the key to efficient teaching of the system, since producing a set of images for a task, and letting the vision system automatically extract the measurements ($\mathbf{y}_i$) is considerably easier than the traditional trajectory specification method used in conventional robot control.

A large number of different visual measures have been proposed for visual feedback, including line length, projected area, area or line length ratios [3], and Fourier descriptors [4]. Mitch Harris in [14] defines a more general "visual similarity" or "closeness" function based on orientation, length, and distance between line segments. Our philosophy has been to use the simplest measures that suffice for the task at hand.

We tried various measures for our perception vectors. As it turned out, very little structural sophistication was needed to obtain perceptual measures that permitted very good performance for a range of positional tasks. The image locations of tracked object features in different cameras (without correspondence) provided satisfactory perceptual features for all the experiments in this paper. The dimension m of our image space $\{\mathbf{y}\}$ then depends on the number of features tracked, one feature in one camera giving two scalar values in a y-vector. The task specifications took one of two forms. Either they specified the desired image locations of various tracked feature points in various cameras, or they specified pairs of tracked feature points that were to be brought into some sort of correspondence with each other (e.g. bringing a rod end to a hole). This latter specification can be used in applications involving a moving camera or an autonomously moving object.

At a higher level, a perceptual action system can utilize a sequence of goal perceptions (feature vectors

if we are using vision), each being a subgoal in a task, thus implementing a cycle *precondition perception* $\rightarrow$ *action* $\rightarrow$ *perception**, where *perception** is a goal perception. This can be seen as an extension of the *precondition perception* $\rightarrow$ *action* used in many subsumption architecture systems [27, 25, 26], where a process is watching sensory inputs for a precondition perception, and when a matching perception occurs, an open loop, canned action is carried out. Our approach extends on the subsumption idea so that the action is no longer open loop, but under continuous monitoring in perceptual space attempting to reach a specific perceptual goal.

4.2 The extended control system for visual manipulation

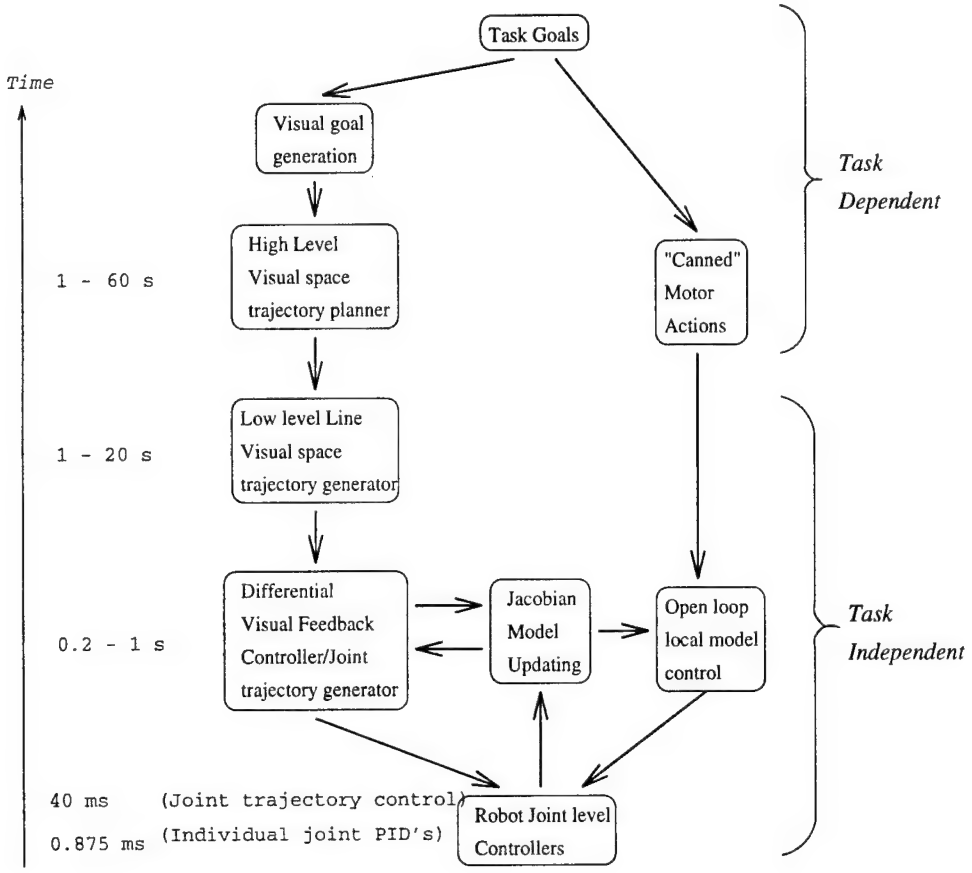


Figure 13: Overall structure of our system, and the time frames each part runs in. Arrows between modules indicate direction of information transfer.

The bare visual feedback controller can be used directly to obtain alignment between features as long as these features are visible. In some cases, however, the initial features may disappear during the course of the manipulation. In the case of object insertion, for example, the object typically moves out of view. In such circumstances, the visual goals directing the operation may not refer directly to the features defining the operation, but to other features whose appearance is functionally related to the defining features. For example, during an insertion operation, the bottom of the object being inserted is no longer visible. However, a gripper is connected (often rigidly) to the inserted object, and hence the configuration of the inserted object can be monitored indirectly by monitoring the configuration of the gripper. Such transitions in the

representation are dictated by the high-level planner. However, it turns out to be useful to define local, visual coordinate systems that specify directions such as "down" or "into the hole". These local visual coordinate systems can then be used both to generate the virtual sketches needed for indirect control, and to define *Canned motor actions*, which are essentially open-loop strategies, pre-defined, in either world vision or robot (joint) space, for effecting some immediate goal such as insertion.

4.2.1 Visual goal generation

Obtaining the visual goals representing relevant goals in a task is a significant issue. Most of our experiments to date, have been performed with the system in a semi-autonomous, or tele-assistance mode, which means that the controlling visual goals are generated with the assistance of an operator. Currently we point out visual goals either by using a mouse in a picture of the scene, or by showing the system an image of the desired pose, and having it extract the features forming the desired perception vector. We have found that supplying real pictures of the desired goal pose yields very accurate positioning, as it assures that the desired pose $\mathbf{y}^*$ is actually physically possible. Here we view the role of the human as that of a vision subroutine, pointing out image locations of the objects requested. In many industrial applications automatic machine vision systems can be used to perform this task.

Currently, we hard code the sequencing of visual goals for each task type. For example in the puzzle demo discussed later in this paper, the high-level sequence for insertion of a puzzle piece (move to piece, pick it up, move near goal, perform insertion, release piece) is provided a-priori. An alternative, still short of full autonomy, would be to learn the high-level description using sensory data taken while a human performs the task, but this requires (sub) task segmentation, which has proved to be a hard problem. Another is to use task level reasoning to figure out how to solve the problem, however this is still some distance in the future.

Our system can also be used as a high level, user friendly, telemanipulation system, having the user specify and sequence what is to be done by pointing out objects to be manipulated in images of the telemanipulation scene, and tagging them with a label describing what is to be done to the object. The manipulations are then carried out autonomously. This significantly lowers the bandwidth in the human-machine interaction, and decouples the human from the kinematics of the actual manipulator used. The above approach can be compared to a system in which human movements are used directly to drive robot joints. This kind of system requires the manipulator to be near anthropomorphic, or requires a significant learning effort on the part of the operator to figure out the manipulator kinematic and dynamics [30].

4.2.2 Trajectory planning

Visual space trajectory planning actually takes place on two levels. The *high level visual space trajectory planner* is task dependent, and deals with making an overall plan of how to get from initial configuration $\mathbf{y}_0$ to the goal $\mathbf{y}^*$, while avoiding obstacles, staying out of singularities, keeping within the convergence range of the controller etc. This planner basically supplies a sequence of visual goals representing important transition points in the task We will describe high level planners for several tasks in the experiments section.

The visual goals provided by the high-level planner may not, however, be adequate for trajectory generation, especially for long paths. The main problem is in cases where the Jacobian changes substantially over the course of the path. The *Low level visual space trajectory generator* described in section 2 further decomposes the trajectory into segments of a length suitable for the differential visual feedback controller.

5 Experiments using visual specification

This section demonstrates the usefulness of visual space specification with three hand-eye tasks that are solved using the self calibrating visual specification and control method described in this paper.

Note: Electronic m-peg videos of all the demonstrations in this section are accessible through the Internet World Wide Web (WWW) pages. Use the menu in:

`http://www.cs.rochester.edu/u/jag/PercAct/videos.html` or

`http://www.cs.chalmers.se/u/jag/PercAct/videos.html`

to select and view the videos corresponding to figures marked **Video #**.

5.1 Visually specified primitives

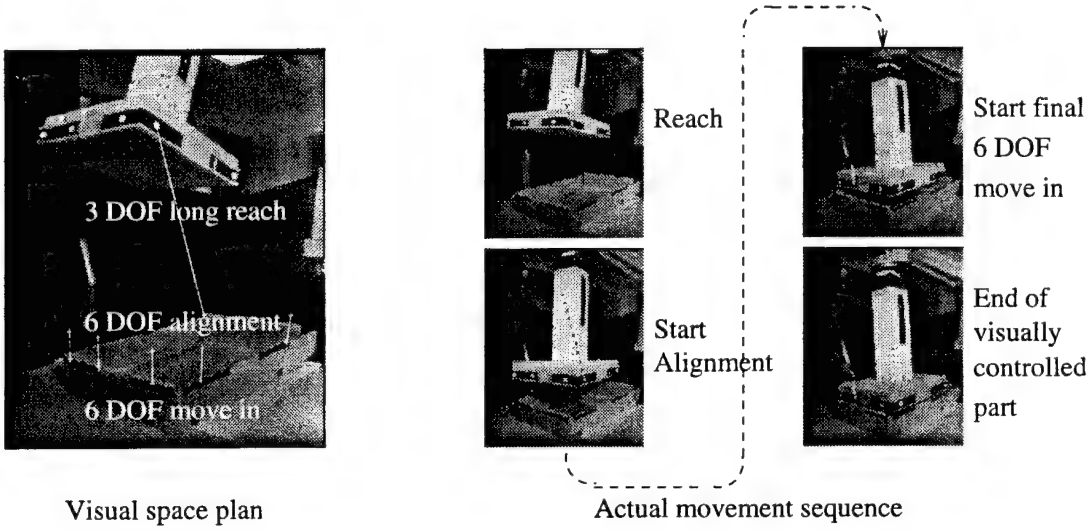


Figure 14: Left: Planning the different phases of an insert type movement consisting of reaching and fine manipulation movements. Right: Performing the planned insertion. **Video 1**

The visual feedback control algorithms provide a means of effecting a visually specified goal while attending to a set of features. Real tasks generally consist of a sequence of goals, and the perception vectors specifying each usually involve different features. There may, however, be certain classes of primitive tasks which make similar use of features, and which re-appear in different applications. We start by identifying and implementing two such primitives which are useful in the general domain of object manipulation. The first primitive class is what we term a transportation move. The basic objective is to bring an object close to a goal position from a distance several times the size of the object. At this stage, the exact orientation and positioning accuracy is not critical. Because transportation is primarily concerned with position in 3-D space, we implement it as a point location problem under 3 DOF control. The second primitive class is fine manipulation of the sort required to perform insertion, alignment, or position a gripper for a pickup. Here accuracy and alignment are important, so we use a 6 DOF controller, which can handle the spatial freedoms of a rigid object.

We can now define a general move as consisting of a transportation phase followed by a fine alignment phase, optionally followed by an insertion or pickup move, which is either performed open loop, or executed

under control of another sensory modality such as touch. Fig. 14 shows an example of an insertion sequence using this general move strategy. The robot first executes a long transportation move to a goal specified by the visual location of a single point in two cameras, then a 6 DOF alignment move to a goal specified by five point locations in in one camera and two in another. Note that we do not need any correspondence between features in the two image, nor do we even have to track the same features. The visual alignment goal is a position just above the insertion hole, and is indicated in the figure by the five + symbols. These are obtained using a local coordinate frame that specifies the visual direction of "up" in the two cameras. This knowledge was, in turn, derived from vertical lines on the object into which the insertion was being performed.

What occurs technically is that at the transition from 3 DOF control to 6 DOF the Jacobian is expanded from a 4×3 to a 14×6 matrix (tracking five feature points in the image shown, and two in the other). The newly added features are expected to behave roughly the same as the old one in response to the 3 trained DOF's, so we can expand the 4×3 Jacobian into a 16×3 estimate. The rest of the DOF's not previously explored are filled with random numbers of magnitude commensurate with the rest of the Jacobian. The function of the alignment is thus twofold: It positions the piece for the precision insertion, and it updates the (now expanded) internal motor-visual Jacobian model to be accurate around the insertion point.

5.2 Combining primitives to tasks: Solving a puzzle

In this demonstration we assemble control primitives from three classes into a program capable of solving a puzzle consisting of the insertion of differently shaped pieces into correspondingly shaped holes. The three modes used are the 3 DOF transportation for moving the pieces around above the table, fine manipulation for alignment and insertion, and open loop manipulations using a local model for the dropping of the pieces inside the box while the pieces are occluded. A minor variation of the transportation primitive, a guarded move, is used for the pickup, to move the manipulator until it touches the puzzle piece to be picked up.

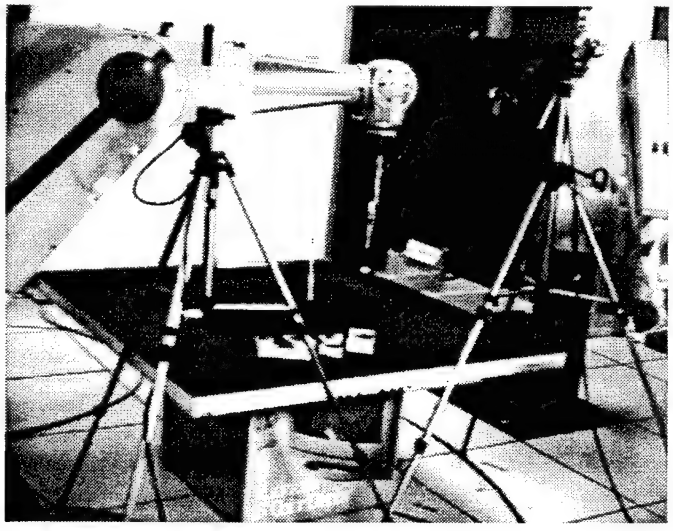


Figure 15: The physical setup in the puzzle demonstration. A Puma robot, equipped with a magnet on a stick is used to pickup, move and insert different shaped puzzle pieces into a box with correspondingly shaped holes. The robot actions are viewed by two uncalibrated cameras, giving visual input to a visual space planner and visual feedback controller

Fig. 15 shows the setup of the puzzle experiment. The cameras are placed fairly arbitrarily. The only

requirements are that they should be able to see some relevant features during the actions, and have a sufficiently different viewpoint to get a reasonable condition number on the motor visual Jacobian. The system does not possess any geometric information about either the cameras or the manipulator, and must solve the task using only visual information.

The dimensions of the puzzle pieces are in the 5 to 10 cm range. The holes are cut about 1mm bigger than the pieces. 1 pixel image offset represents from 2 to 0.5 mm physical movement for a movement in the image plane, depending on where in the image the movement is made. The constraints on the problem are about as tight as can be solved, given limits inherent to the vision system and manipulator.

The basic insert operation consists of four phases. In the first, the robot moves the manipulator to pick up a puzzle piece. the second phase is a long reach or transportation move to bring the puzzle piece to a location approximately above the hole. A fine manipulation move then aligns the puzzle piece for insertion. Finally, an open loop process inserts the piece and scrapes it off against the side of the hole.

The two first frames of fig. 16 show the visual space plans made to pickup and insert the leftmost puzzle piece. An estimate of "up" in image space is known to the algorithm. It is given once by inserting a long vertical, easily identified object, here a pencil, in the image. We have not implemented an object recognition system for this application. We do not anticipate that it would be hard to do, but it was not the main aim of our work. Instead while the program runs, it shows the user images of the scene, and asks the user to point out with the mouse where in the image a puzzle piece is located, and the corners of the slot it is supposed to be put into. After that the system uses the "up" information to make the visual plan, and starts moving according to the plan. The transportation and fine manipulation movements are executed as described in the previous section.

In this application, the robot "grasps" objects using a pair of magnets: one in the object to be picked up, and another in the tip of the manipulation tool. The guarded pick-up move is implemented by using flexibility in the tool. When the manipulator touches a puzzle piece, the magnet, whose visual position is monitored by a feature tracker, will stop moving downward. This is detected, and the movement is stopped. Technically the contact detection is done by watching the model error (difference between predicted and actual movement), and terminating the guarded move on a steep increase in model error. After pickup, the piece is transported to a goal location above the appropriate hole, determined using the image space estimate of up. Visual feedback during the fine manipulation is obtained by tracking the lower corners of the puzzle piece that are visible in each camera. The goal for the fine manipulation is to bring these corners to the corresponding corners of the hole, previously pointed out by the operator.

That is as far as we get with visual feedback. The rest of the insertion and drop off is done using the local model of the vision to motor space mapping provided in the 3×4 Jacobian estimated during the long reach movement. We already know the vector representing world space (physical) up in visual coordinates ($\hat{\mathbf{up}}$). To put the piece in we solve $-a\hat{\mathbf{up}} = J\Delta\mathbf{x}$ where a is some suitable fraction of the height of the box, measured in pixels in image space. To deposit the puzzle piece in the box, we release it from the magnet by scraping it on the sides of the hole. For this we need estimates of sideways movement. We can't get very exact such without trying to identify sideways going lines. For the scraping it turns out to be enough just to use two lines in vision space orthogonal to our "up" vector.

When the piece is deposited the user is requested to click on a new piece and the process is repeated. Thus used in tele-assisted, or semi-autonomous mode, the robot was able to solve the puzzle. The last frame in the puzzle sequence, fig. 16 shows an actual trajectory (solid) compared to the planned (dotted).

Our current implementation has several limitations. First, because the tracker needs fairly good corners, we can't handle complex curved shapes. The system also has difficulty with large rotations of the pieces. In order to handle this, the visual planner needs to make visual plans corresponding to circles in world space (the straight line approximation is no longer good enough). As features become occluded during large rotations

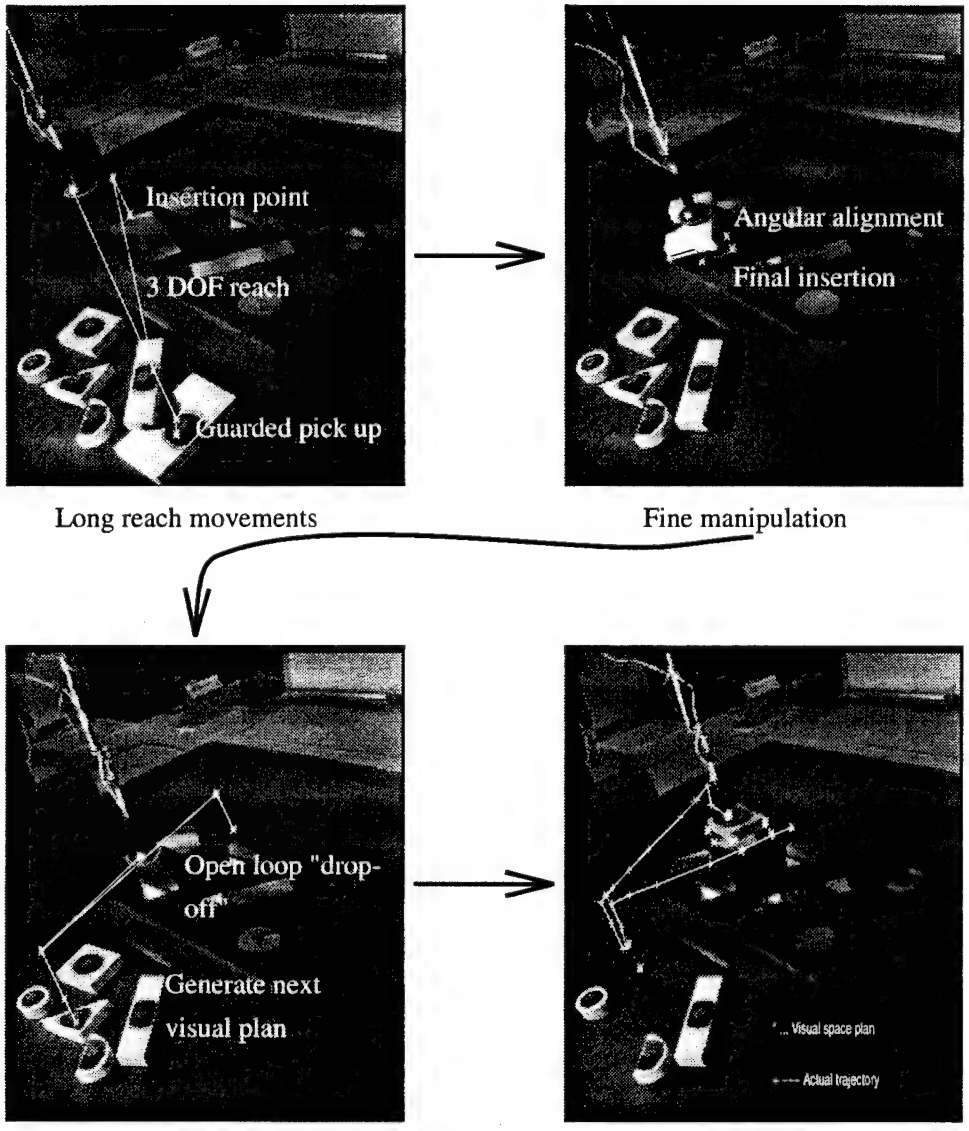


Figure 16: Different modes of control are used in the puzzle laying process **Video 2** and **Video3**

we we need to dynamically take them out from the Jacobian model, and add new features to replace them. This is relatively easy to do by training the new features for a few ($\approx \#DOF$ depending on the filtering) Jacobian updating cycles, without having them affect the movement, and then put them into the feedback loop. The harder part is a correspondence problem: For each one of the new features a goal value has to be found, and asking the user becomes tedious. Some higher-level recognition process could be used here.

5.3 12 DOF Control of a Non-rigid Foam Beam

High DOF control problems involving manipulation of non rigid objects are very hard to solve with traditional model based robot control paradigms. The modeling of the problem can be messy or completely impractical. Even in cases where an analytic model can be provided it is often not analytically invertible, which makes design of a control system difficult. An adaptive differential visual feedback based controller, on the other hand, does not need an exact a-priori model. Instead the robot learns and refines successive linear models of the transfer function while performing its task in the real environment.

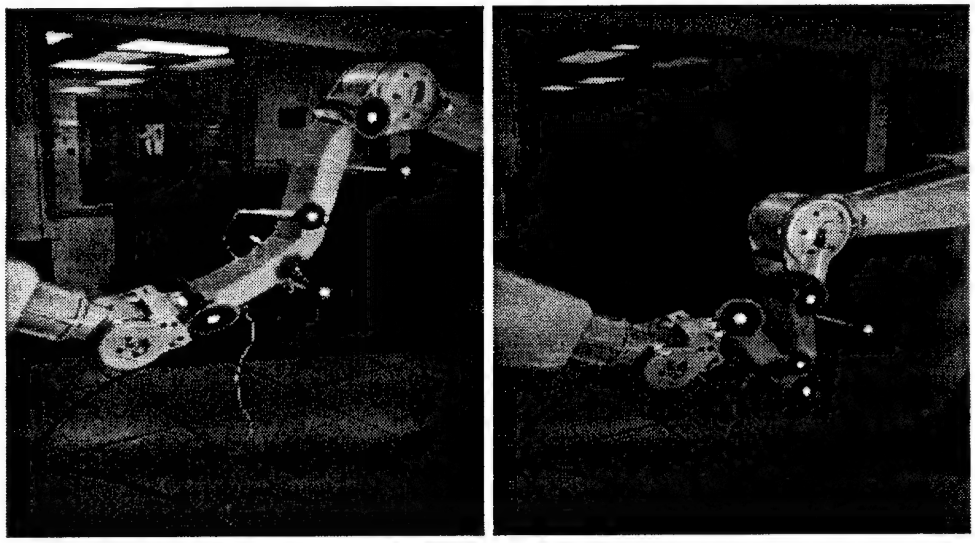


Figure 17: Left: Initial configuration of two manipulators holding a piece of foam between them. Right: The goal is given as an image of the desired pose of the foam, from which the desired positions of the "features", here the white dots, are extracted.

This demonstration uses two 6 DOF PUMA manipulators, one attached to each end of a piece of flexible packing material foam. The attachment is rigid, so the arms can exert torques as well as forces to the beam. The object of the system is to bend or fold the beam into a specified shape.

The goal is specified by providing goal an image of a folded piece of foam. This is done to make sure that the desired configuration is actually attainable. (The foam can be manipulated in a theoretically infinite, and practically very large number freedoms. We can only control 12 with the manipulators, so many configurations would not be attainable.) In practice, we used the robot teach pendants to manually put the foam into the shape we desire, however the control algorithm never gets any input about the pose of the robots needed to do this, just a representation of the foam shape, given in image space from two cameras. In the demonstration we start out with the foam just barely compressed between the two robots end effectors as seen in the left image in 17. We originally intended to track the foam shape using snakes

[23, 24]. This proved hard, so we reverted to our convolution trackers, tracking a sparse representation of the foam contour using the tracking targets shown attached to the foam in fig 17.

From a transfer function point of view this is a hard problem. To start with, the transfer function is highly nonlinear. We also start from a nearly singular state (nearly straight beam). (The singularities are not just robot singularities, but all singularities in the full transfer function from robot control space to visual space) As evidence, we note that the condition number of the visual motor Jacobian improves by nearly a factor of 20 between the initial state and the goal.

Figure 18: Hitting joint limits in a robot singularity caused failure in this "non way point" run. **Video 4**

This problem is hard enough that running the system using only the final goal position did not work. The problem is that the Jacobian changes so much from the initial to the final configuration that aiming directly for the goal from the start leads the system into unstable and sometimes singular configurations where it could become trapped. An example of this is shown in Fig. 18 where joint 6 and 4 are lined up giving a singular state. The difficulty was easily resolved by giving the system a few way points (in this case two), which served to guide the algorithm around numerical (and mechanical) difficulties. In numerical analysis this is a standard method of improving convergence. With this constraint, the system was able to avoid getting onto an ill-conditioned path, and fold the beam into the goal configuration with an average end point accuracy in image space of about one pixel. This "way point" solution is illustrated in fig. 19.

6 Conclusions

This paper has introduced the concept of perceptual actions as a schema for high level visual control of manipulation. We have also introduced a powerful method of adaptive visual feedback control, and coupled it into this framework. This allowed us to demonstrate visual control of manipulation for some difficult, non-linear problems. We described how the concepts of perceptual action and the visual goals can be used as a specification language for visually guided manipulation, and illustrated the concept by implementing a semi-autonomous robotic system that was able to perform complex visually guided tasks.

References

[1] Sanderson A. C. Weiss L. E. "Adaptive visual servo control of robots" In *Robot Vision* Ed: Pugh A. IFS 1988

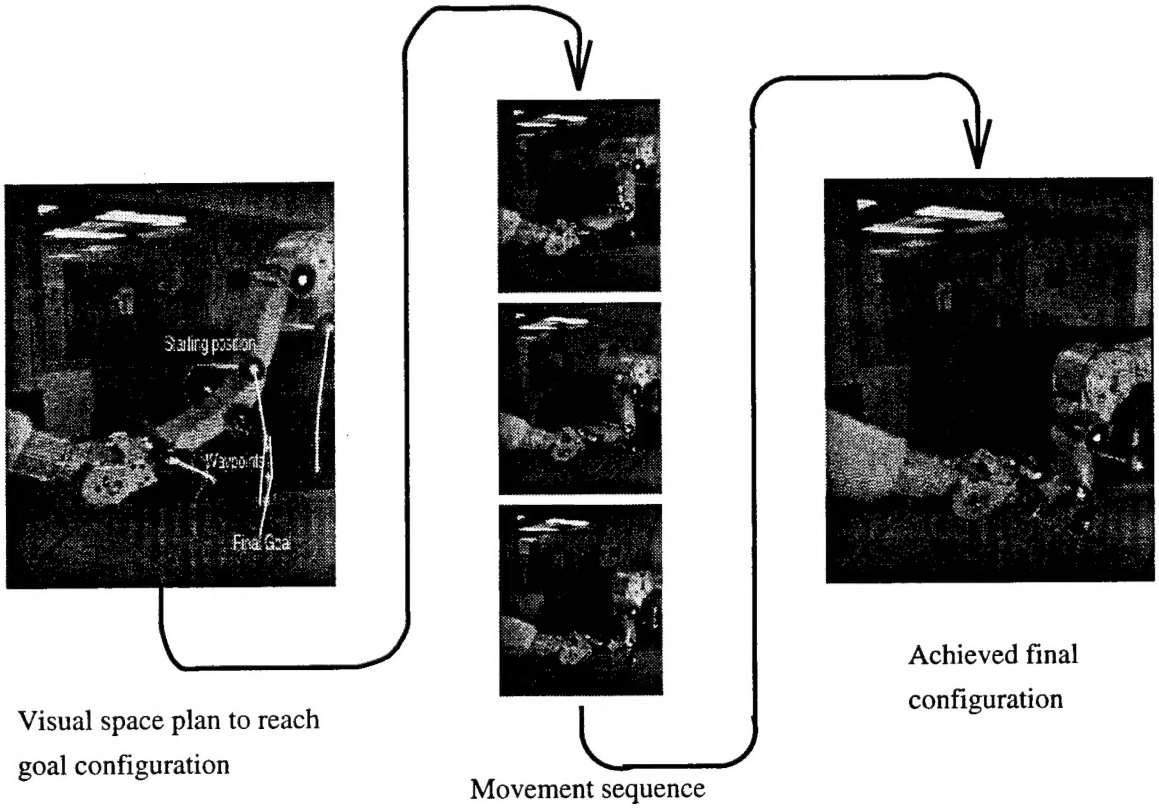


Figure 19: The carrying out of the foam folding task **Video 5**

[2] Weiss L. E. "Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach" PhD thesis, CMU 1984

[3] Weiss L. E. Sanderson A. C. Neumann C. P. "Dynamic Sensor-Based Control of Robots with Visual Feedback" *J. of Robotics and Automation* v. RA-3 Oct 1987

[4] Feddema J. T. Mitchell O. R. "Vision Guided Servoing with Feature Based Trajectory Generation" *IEEE Trans. on Robotics and Automation* v 5 nr 5 p 691-670 1989

[5] Feddema J. T. Lee G. C. S. "Adaptive Image Feature Prediction and Control for Visual Tracking with a Hand-Eye Coordinated Camera" *IEEE Trans. on Systems, Man and Cybernetics*, vol 20, no 5 1990

[6] Feddema J. T. Lee G. C. S. Mitchell O. R. "Weighted selection of Image Features for Resolved Rate Visual Feedback Control" *IEEE Trans. on Robotics and Automation*, v 7 nr 1 p 31-47 1991

[7] Feddema J. T. Lee G. C. S. Mitchell O. R. "Model-Based Visual Feedback Control for a Hand-Eye Coordinated Robotic System" *Computer*, Aug 1992

[8] Papanikolopoulos N. P. Khosla P. K. "Adaptive Robotic Visual Tracking: Theory and Experiments" *IEEE Trans. on Automatic Control* Vol 38 no 3 March 1993

[9] Chongstitvatana P. Conkie A. "Behaviour Based Assembly Experiments using Vision Sensing" DAI TR #466, U of Edinburgh 1990

[10] Conkie A. Chongstitvatana P. "An Uncalibrated Stereo Visual Servo System" DAI TR #475, U of Edinburgh 1990

[11] J. Y. Zheng Q. Chen S. Tsuji "Active camera guided manipulation" In *Proc. of 1991 IEEE Int Conference of Robotics and Automation*

[12] Bennet D. Geiger D. Hollerbach J. "Autonomous Robot Calibration for Hand-Eye Coordination" *Int. J. of Robotics Research* v 10 nr 5 1991.

[13] Wijesoma S. W. Wolfe D. F. H. Richards R. J. "Eye-to-Hand Coordination for vision guided Robot Control Applications" *Int. J. of Robotics Research*, v 12 No 1 1993

[14] Harris M. "Vision Guided Part Alignment with Degraded Data" DAI TR #615, U of Edinburgh 1993

[15] Hager G. Chang W.C. Morse S. "Robot Feedback Control Based on Stereo Vision: Towards Calibration-Free Hand-Eye Coordination" TR # 992 Yale 1993

[16] Hollinghurst N. Cipolla R. "Uncalibrated Stereo Hand-Eye Coordination" *British Machine Vision Conference* 1993

[17] Jägersand M. "Perception level control for uncalibrated hand-eye coordination and motor actions" Areapaper, University of Rochester May 1994. Forthcoming as TR.

[18] Corke P. I. "Visual control of robot manipulators - a review" In *Visual Servoing*, World Scientific 1993.

[19] Corke P. I. "High-Performance Visual Closed-Loop Robot Control" PhD thesis U of Melbourne July 1994.

[20] Hosoda K. Asada M. "Versatile Visual Servoing without Knowledge of True Jacobian" *IROS* Aug 1994.

[21] W. Z. Chen U. A. Korde S. B. Skaar "Position Control Experiments Using Vision" *Int. Journal of Robotics Research*, v13 No 3 p199-208, 1994

[22] B. H. Yoshimi P. K. Allen "Active, Uncalibrated Visual Servoing" Submitted *1995 Int. Conference on Robotics and Automation*

[23] Terzopoulus D. Szeliski R. "Tracking with Kalman Snakes" In *Active Vision*, Ed Blake, Yuille MIT Press 1992.

[24] Terzopoulus D. Szeliski R. "Dynamic Contours: Real time active splines" In *Active Vision* Ed Blake, Yuille MIT Press 1992.

[25] Brooks R. "A Robust Layered Control System For A Mobile Robot" *IEEE J. Robotics and Automation*, v RA-2 nr 1 March 1986

[26] Brooks R. "Intelligence Without Reason" AI memo nr 1293 MIT 1991

[27] Connell J. "Controlling a mobile robot using partial representations" *SPIE 91 Mobile Robots* pp 34-45 1991

[28] Ikeuchi K. Suehiro T. "Towards an Assembly Plan from Observation" *Proc. Robotics and Automation* 1992

[29] Kuniyoshi Y. Inaba M. Inoue H. "Seeing, Understanding and Doing Human Task" *Proc. Robotics and Automation* 1992

[30] P. Pook, D. H. Ballard "Teleassistance: Contextual guidance for autonomous manipulation" *Proc. of AAAI, Aug 1994.*

[31] Lloyd J. "The Multi-RCCL Users Guide" *ICRA Workshop on visual servoing* 1994.

[32] G. A. Geist, V. S. Sunderam "Network Based Concurrent Computing on the PVM System" TR Oak Ridge National Laboratory, 1993

[33] Fletcher R. *Practical Methods of Optimization* Chichester second ed. 1987

[34] Gustafsson I. *Tillämpad Optimeringslära* Kompendium, Institutionen för Informationsbehandling, Chalmers 1991

[35] Nelson R. "Vision a Intelligent Behavior-An introduction to Machine Vision at the University of Rochester" *IJCV* 7:1 p 5-9 Kluwer 1991